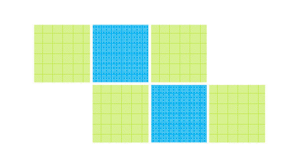
PHP Coding Standards





إعداد / ماجد المليحاني







المعايير القياسية لكتابة الشيفرة المصدرية

PHP Coding Standards

إعداد / ماجد المليحاني

الإصدار الأول

١٤٣٢هـ - ٢٠١١م

المحتويات

مقدمة :

أن الحمد لله ، نحمده ونستعينه ونستهديه ، ونعوذ بالله من شرور أنفسنا وسيئات أعمالنا ، من يهده الله فلا مضل له ، ومن يضلل فلا هادي له ، وأشهد ألا اله إلا الله وحده لا شريك واشهد أن محمد عبده ورسوله صلى الله عليه وسلم ، وعلى آله وأصحابه وأتباعه إلى يوم الدين ... أما بعد

إن بعضاً من المبرمجين وخصوصا المبتدئين منهم لا يتبع أي معايير قياسية أثناء كتابته للشيفرة المصدرية أو ما يعرف بالكود البرمجي (Source code). ولقد تم إعداد هذا الكتيب لتسليط الضوء على أهمية المعايير القياسية للشيفرة المصدرية (Coding Standards) والتعرف على أهم هذه المعايير.

فكما نعلم جميعا أنه بإمكان أي شخص كتابة الشيفرة المصدرية (Source code) . ومع قليل من الخبرة بالإمكان عمل العديد من البرامج . ولكن لعمل ذلك بالشكل الصحيح نحتاج إلى الكثير من التدريب والعمل.

إن معايير كتابة الشيفرة المصدرية توحد الطريقة التي يجب إتباعها عند كتابة هذه الشيفرة. فبدلا من أن يكتب كل مطور الشيفرة بأسلوبه الخاص يتم توحيد طرق قياسية لكتابتها وفق معايير معينة. والهدف من هذه المعايير هو جعل الشيفرة المصدرية أكثر قابلية للقراءة.

هذه المعايير مهمة جدا خاصة للمشاريع التي يتم تطويرها من قبل العديد من المطورين. فهي تساعد على كتابة الشيفرة المصدرية بجودة عالية وأخطاء اقل . كما أنها تسهل من عملية صيانتها.

إن النقطة الأساسية من معايير كتابة الشيفرة المصدرية ليس للقول بان طريقة معينة أفضل من الأخرى ولكن لتنظيم الشيفرة بطريقة تريحك كثيراً بالعمل وكذلك من يعمل بعدك على نفس الشيفرة.

هناك أكثر من طريقة قياسية لكتابة الشيفرة. وسوف نتعرف في هذا الكتيب إن شاء الله على معايير مكتبة **PEAR** لطريقة كتابة الشيفرة المصدرية لكونها الأكثر شعبية. حيث أن هذه المعايير معروفة ومقبولة أكثر من غيرها. وتعتبر الأساس للعديد من المشاريع مفتوحة المصدر مثل **Zend Framework - Solar – Horde - Drupal** . ومعتمدة من قبل الكثير من المطورين.

ويجب التنويه إلى أن هذه المعايير قابلة للتغير . كما أننا غير ملزمين تماماً بإتباعها. وعدم الالتزام بها وتطبيقها أثناء كتابة الشيفرة لا يؤثر على طريقة تنفيذ البرامج .

ويبقى العديد من الجوانب التي تخص هذه المعايير بحاجة إلى المزيد من التوسع حيث أنني هنا حاولت تغطية أهما فيما يخص كتابة الشيفرة المصدرية بشكل عام . وليعذرني القارئ الكريم إذا وجد بعض الركاكة والنقص هنا أو هناك حيث أن المبتغى هو رضوان الله تعالى **أولا** وتسليط الضوء على أهمية المعايير القياسية للشيفرة المصدرية ومعرفة أهم معايير مكتبة **PEAR** لكتابتها **ثانيا**.

كما يسعدني ويشرفني استقبال جميع الملاحظات والاستفسارات حول هذا الإصدار على البريد الإلكتروني **majed@modernsys.net** .

" والله ولي التوفيق "

ماجد سليمان المليحاني

لمن هذا الكتيب ؟

هذا الكتيب يتناول أهم معايير مكتبة **PEAR** لكتابة الشيفرة المصدرية لذلك هو موجه بالدرجة الأولى لجميع مبرمجين لغة **php** . نحن هنا لن نتطرق لمكتبة **PEAR** بشكل مفصل وإنما نتعرف فقط على أهم معاييرها في كتابة الشيفرة . ولمن يرغب بالاطلاع على المزيد حول مكتبة **PEAR** يمكن زيارة موقعها الرسمي على الرابط التالي **http://pear.php.net**

ما هي مكتبة PEAR ؟

إن كلمة **PEAR** هي اختصار للعبارة **PHP Extension and Application Repository** مخزن ملاحق وتطبيقات **php** . وبعبارة أخرى ، نظام توزيع وإطار عمل لمكونات **php** التي يمكن إعادة استخدامها. حيث تحتوي على مكونات تم برمجتها مسبقا ، والتي يمكن استخدامها لجعل كتابة الشيفرة أكثر سهولة في **php** . توفر هذه المكتبة وحدات نمطية **(modules)** مفتوحة المصدر تقوم بتوزيعه وصيانتها . كما توفر **معايير قياسية** لكتابة شيفرة هذه الوحدات النمطية.

حقوق الطبع :

محتويات الكتيب :

بإذن الله تعالي سوف نتطرق إلى أهم معايير مكتبة **pear** في كتابة الشيفرة المصدرية المتمثلة فيما يلي :

- الإزاحة وطول السطر - Indenting and Line Length
- بنى التحكم - Control Structures
- استدعاء الدوال - Function Calls
- تعريف الدوال - Function Definitions
- تعريف الفئات (الأصناف) - Class Definitions
- المصفوفات - Arrays
- التعليقات - Comments
- تضمين الكود من ملف خارجي - Including Code
- وسم بي اتش بي - PHP Code Tags
- التسمية - Naming Conventions

كما سوف نقوم ببعض الممارسات و التدريبات على كتابة شيفرة قياسية مثالية . ويجب التنبيه على أن الشيفرات المذكورة في هذا الكتيب إنما هي للتوضيح فقط وليس لها أي استخدام برمجي فلا ترهق نفسك في محاولة فهم الشيفرة حرفيا وإنما ركز على فكرة الشيفرة بشكل عام.

١. الإزاحة وطول السطر - Indenting and Line Length

يقصد بالإزاحة الفراغ في بداية السطر البرمجي ، ويقصد بطول السطر عدد الأحرف لكل سطر برمجي بما فيها الفراغات التي بين الحروف. أما المعايير التي يجب إتباعها فيما يخص الإزاحة وطول السطر هي كتالي :

- يجب أن تكون الإزاحة بمقدار أربع أحرف (فراغات) بدون استخدام الزر Tap .
- يجب أن يكون طول السطر البرمجي بين ٧٥-٨٥ حرف.

مثال لاختصار طول السطر :

طول هذه السطر البرمجي أكثر من ١٢٠ حرف

```
list($name, $age, $gender, $identity) = array(get_values($name), get_values($age),
get_values($gender), get_values($identity:((
```

يمكن اختصاره بالشكل التالي :

```
list($name, $age, $gender, $identity) = array(
    get_values($name),
    get_values($age),
    get_values($gender),
    get_values($identity)
);
```

حيث قمنا بوضع عناصر المصفوفة على عدة اسطر مع إزاحتها إلى الداخل بمقدار أربع أحرف وقوس نهاية المصفوفة في سطر مستقل بدون إزاحة. وهذا يجعل من الشيفرة أكثر قابلية للقراءة .



٢. بنى التحكم - Control Structures

وتشمل كل من if, for, while, switch وغيرها من بنى التحكم.

وهذا مثال لعبارة if لأنها الأكثر تعقيدا بينهم :

```
if ((condition1) || (condition2)) {
    action1;
} elseif ((condition3) && (condition4)) {
    action2;
} else {
    defaultaction;
}
```

- يجب أن يكون هناك فراغ واحد (بمقدار حرف واحد فقط) يفصل بين عبارة التحكم وقوس بداية جملة التحكم وذلك للتفريق بنها وبين استدعاء الدوال وكذلك فراغ واحد بين علامات المقارنة والشروط . مثال :

```
if (($age == 18) && ($gender == 'MALE')) {
    //... code goes here
}
```

- يفضل دائما استخدام الأقواس المتعرجة { } حتى في الحالات التي يمكن الاستغناء عنها أو التي يكون استخدامها اختياري . فاستخدامها يزيد من قابلة قراءة الشيفرة ويقلل من حدوث الأخطاء المنطقية عند إضافة سطر لجملة التحكم.

- تكتب عبارة التحكم في سطر لوحدها وينتهي السطر بعد قوس بداية التعليمة . مثال :

```
if ((condition1) || (condition2) ) { // نهاية السطر
```

- يجب إزاحة الشيفرة داخل أقواس التعليمة { } بمقدار أربع أحرف . ويجب كتابة قوس نهاية التعليمة في سطر مستقل. مثال :




```
if ((condition1) || (condition2)) {
    //... إزاحة الشيفرة بمقدار أربع أحرف
}
```

- يجب أن تستخدم العلامات && و || بدلا من كلماتها AND و OR

مثال على switch :

```
switch (condition) {
case 1:
    action1;
    break;

case 2:
    action2;
    break;

default:
    defaultaction;
    break;
}
```

٣. استدعاء الدوال - Function Calls

- يجب استدعاء الدالة بدون فراغ بين أسم الدالة وقوس البداية. مثال :

```
get_values()
```

- عند كتابة معطيات (باراميترات) الدالة يجب الفصل بينها بفراغ واحد فقط ويجب أن يكون هذا الفراغ بعد الفاصلة وليس قبلها. مثال :

```
($name, $age, $gender)
```

- لا يجب أن يكون هناك فراغ بين أول معطى وبين قوس بداية الدالة وكذلك بين آخر معطى وقوس نهاية الدالة. مثال لدالة مثالية :

```
get_values($name, $age, $gender);
```

- يجب أن تعيد الدالة قيمة باستخدام التعليمة **return** .
- إزاحة علامة المساواة عندما يكون هناك متغير ذو اسم قصير وآخر ذو اسم طويل وذلك لزيادة قابلية قراءة الكود . مثال :

```
$short         = foo($bar);
$long_variable = foo($baz);
```



- ويمكن كسر قاعدة إزاحة علامة المساواة إذا كان أحد المتغيرات طويل جدا. مثال :

```
$short = foo($bar);
$thisVariableNameIsVeeeeeeeeeeryLong = foo($baz);
```

- ولزيادة قابلية قراءة الشيفرة عند استدعاء الدوال يمكن عمل إزاحة لمعطيات الدالة كما في المثال التالي :

```
$this->callSomeFunction('param1',     'second',        true);
$this->callSomeFunction('parameter2', 'third',         false);
$this->callSomeFunction('3',          'verrrrrrylong', true);
```

- من المستحيل أحيانا أن يكون طول السطر البرمجي للدالة بين ٧٥ - ٨٥ حرف وخصوصا في حالة وجود معطيات كثيرة للدالة لذلك يمكن وضع المعطيات في أكثر من سطر كما يلي :

```
$this->someObject->subObject->callThisFunctionWithALongName(
    $parameterOne, $parameterTwo,
    $aVeryLongParameterThree
);
```

- علامة المساواة (=) : يمكن تقسيم علامة المساواة على عدة اسطر إذا تجاوزت المعاملات الحد المسموح به لطول السطر البرمجي ويجب أن تكون علامة المساواة في السطر التالي مع إزاحة بمقدار أربع فراغات. مثال :

```
$GLOBALS['TSFE']->additionalHeaderData[$this->strApplicationName]
    = $this->xajax->getJavascript(t3lib_extMgm::siteRelPath('nr_xajax'));
```

٤. تعريف الدوال - Function Definitions

- يجب أن يكون قوس بداية الدالة وقوس نهاية الدالة في سطر لوحده ويجب إزاحة الشيفرة بداخل أقواس الدالة بمقدار أربع فراغات. مثال :

```
function fooFunction()
{
    //... code goes here
}
```

- يجب أن تعيد الدالة قيمة باستخدام التعليمة **return** .
- المعطيات (البارامبترات) التي تكون لها قيم افتراضية تكون في آخر قائمة المعطيات. مثال :

```
($arg1, $arg2 = '')
```

- مثال لدالة قياسية :

```
function fooFunction($arg1, $arg2 = '')
{
    if (condition) {
        statement;
    }
    return $val;
}
```

- مثال لدالة قياسية أخرى :

```
function connect(&$dsn, $persistent = false)
{
    if (is_array($dsn)) {
        $dsninfo = &$dsn;
    } else {
        $dsninfo = DB::parseDSN($dsn);
    }
```




```
    if (!$dsninfo || !$dsninfo['phptype']) {
        return $this->raiseError();
    }

    return true;
}
```

- في حالة وجود معطيات كثيرة للدالة وتجاوزت الدالة الحد المسموح به في السطر البرمجي يمكن وضع المعطيات في أكثر من سطر. حيث يمكن وضع بعض المعطيات في نفس السطر مع اسم الدالة إذا كان هناك مساحة كافية والباقي في الأسطر التالية مع عمل إزاحة بمقدار أربع فراغات. كما يجب وضع قوس نهاية معطيات الدالة وقوس بداية الدالة في سطر جديد

مثال :

```
function someFunctionWithAVeryLongName($firstParameter = 'something', $secondParameter = 'booooo',
    $third = null, $fourthParameter = false, $fifthParameter = 123.12,
    $sixthParam = true
) {
    //... code goes here
```

٥. تعريف الفئات (الأصناف) - Class Definitions

- يجب أن يكون قوس بداية الصنف وقوس نهاية الصنف في سطر لوحده ويجب إزاحة الشيفرة بداخل أقواس الصنف بمقدار أربع فراغات. مثال :

```
class Foo_Bar
{

    //... code goes here

}
```

٦. المصفوفات – Arrays

- عند تعريف مصفوفة على عدة اسطر يتم إضافة الفاصلة حتى بعد القيمة الأخيرة وهذا يعتبر جملة صحيحة في لغة بي اتش بي (valid PHP syntax) . مثال :

```
$some_array = array(
    'foo'   => 'bar',
    'spam'  => 'ham',
);
```

٧. التعليقات – Comments

- يجب استخدام الشرطة المائلة مع النجمة لكتابة تعليق يحتوي على أكثر من سطر. مثال :

```
/**
 * More than one lines
 * تعليق يحتوي على أكثر من سطر
 * ....
*/
```

- يجب استخدام الشرطة المائلة المكررة لكتابة تعليق من سطر واحد وعدم استخدام الرمز# .

مثال :

```
// single Line
// تعليق في سطر واحد
```

٨. تضمين الكود من ملف خارجي – Including Code

- عند تضمين كود من ملف خارجي بشروط معينة استخدم التعليمة include_once . مثال :

```
$db_type = get_db_type();
if ($db_type == 'mysql') {
    include_once('mysql.php');
}
```

- عند تضمين كود من ملف خارجي بدون أي شروط استخدم التعليمة require_once .

مثال:

```
require_once('header.php');
```

٩. وسم بي اتش بي – PHP Code Tags

- يجب استخدام الوسم (النمط) الطويلة <?php ?> وليس الوسم القصير <? ?>

١٠. التسمية – Naming Conventions

يجب التنبيه إلى أنه تختلف تسمية المتغيرات والدوال التي تكون خارج الأصناف (classes) قليلا عن التسمية داخلها.

- يتم تسمية المتغيرات بأحرف صغيره ويفصل بين الكلمات في اسم المتغير بشرطة سفلية.

```
$db_type = 'mysql';
```

- يتم تسمية الثوابت بأحرف كبيرة ويفصل بين الكلمات في اسم الثابت بشرطة سفلية.

```
define ('SITE_PATH', $sitepath);
```

- يتم تسمية الدوال بأحرف صغيرة ويفصل بين الكلمات في اسم الدالة بشرطة سفلية.

```
get_db_type();
```

- يتم تسمية الأصناف (classes) بحرف كبير في بداية كل كلمة من أسم الصنف ويفصل بين الكلمات في بشرطة سفلية. كما يجب أن يكون للاسم دلالة وصفية . وتجنب اختصار الاسم قدر الإمكان . مثال :

```
HTML_Upload_Error
HTML_Template
Log
```

- يتم تسمية الدوال والمتغيرات داخل الصنف بحرف كبير في بداية كل كلمة من أسم ماعدا الكلمة الأولي مثال :




```
// أسماء متغيرات
$counter
$clearCache

// أسماء دوال
getData()
connect()
buildSomeWidget()
```

- يجب وضع شرطة سفلية قبل اسم الدوال الخاصة (Private) في داخل الصنف. مثال :

```
_sort()
_initTree()
```



بعض الممارسات على كتابة شيفرة قياسية :

لو نظرنا إلى الكود التالي :

```
$name='Majed';
$country='Saudi Arabia';
$city='Madinah';
$site='www.yamamah.org';
$email='mr_amri@hotmail.com'
$data='My name is '.$name.' and I am from '.$country.' - '.$city.
$contact='You can contact me through my site '.$site.' or my email '.$email;
if($name=='Majed'){ echo $data.'<br />'.$contact;}
```

يمكن كتابته بصورة أفضل ليصبح أكثر قابلية للقراءة كما يلي :

```
$name    = 'Majed';
$country = 'Saudi Arabia';
$city    = 'Madinah';
$site    = 'www.yamamah.org';
$email   = 'mr_amri@hotmail.com'
$data    = 'My name is '.$name.' and I am from '.$country.' - '.$city;
$contact = 'You can contact me through my site '.$site.' or my email '.$email;

if ($name == 'Majed') {
    echo $data.'<br />'.$contact;
}
```

حيث قمنا بإزاحة علامة المساواة بين المتغيرات وذلك لوجود متغيرات ذو اسم قصير وأخرى ذو اسم طويل. وكذلك أضفنا فراغ واحد (بمقدار حرف واحد فقط) يفصل بين عبارة التحكم وقوس بداية جملة **if** . وأيضا تم إزاحة الشيفرة داخل أقواس التعليمة **if** (الأقواس المتعرجة {}) بمقدار أربع أحرف.



ولو نظرنا إلى المثال التالي:

```
if ($foo) {
    $bar = 1;
}
if ($spam) {
    $ham = 1;
}
if ($pinky) {
    $brain = 1;
}
```

يكون أسهل في القراءة إذا أضفنا سطر فارغ بين كل جملة من جمل **if** كما يلي :

```
if ($foo) {
    $bar = 1;
}

if ($spam) {
    $ham = 1;
}

if ($pinky) {
    $brain = 1;
}
```

لو أردنا تقسيم عبارة **if** التالية على عدة اسطر

```
if ($condition1 || $condition2 && $condition3 && $condition4) {
    //... code goes here
}
```

يمكن إعادة كتابتها بالشكل التالي :

```
if (   $condition1
    || $condition2
    && $condition3
    && $condition4
) {
    //... code goes here
}
```



و أفضل الحالات بالطبع هو عندما لا يكون هناك حاجة إلى تقسيم السطر البرمجي لعدة اسطر.

عندما تكون الجملة الشرطية طويلة بما يكفي ويجب تقسيمها إلى عدة اسطر يكون من الأفضل تبسيطها. وفي مثل هذه الحالات يمكنك التعبير عن الشروط كمتغيرات ومقارنة هذه المتغيرات في العبارة الشرطية **if** بهذه الطريقة يمكن تقسيم الشرط إلى مجموعات اصغر. ولتوضيح ذلك بشكل اكبر يمكن إعادة كتابة الشيفرة السابقة بالشكل التالي :

```
$is_foo = ($condition1 || $condition2);
$is_bar = ($condition3 && $condtion4);

if ($is_foo && $is_bar) {
    //... code goes here
}
```

كما يمكن استخدام نفس هذه القاعدة مع معامل الشرط الثلاثي (**Ternary operators**) إذا يمكن التقسيم إلى عدة اسطر مع الحفاظ على علامة الاستفهام و النقطتين في بداية السطر . مثال :

```
// علامة الاستفهام و النقطتين في نفس السطر
$a = $condition1 && $condition2
    ? $foo : $bar;

// علامة الاستفهام في سطر و النقطتين في سطر مستقل
$b = $condition3 && $condition4
    ? $foo_man_this_is_too_long_what_should_i_do
    : $bar;
```

للحفاظ على قابلية القراءة للدوال ومعطياتها من الحكمة إعادة قيمة مبكرا عند تطبيق شرط بسيط في بداية الدالة . لو نظرنا إلى المثال التالي :

```
function foo($bar, $baz)
{
    // عند تحقق الشرط
    if ($foo) {
        // يفترض
        // أن
        // يكون
        // هنا
        // شيفرة
        // خاصة
        // تتعامل
        // مع
        // معطيات
        // الدالة
        return $calculated_value;
    } else {
        return null;
    }
}
```

من الأفضل إعادة القيمة null مبكراً إذا لم يتحقق الشرط كما يلي :

```
function foo($bar, $baz)
{
    // عند عدم تحقق الشرط
    if (!$foo) {
        return null;
    }

    // عند تحقق الشرط
    if ($foo) {
        // يفترض
        // أن
        // يكون
        // هنا
        // شيفرة
        // خاصة
        // تتعامل
        // مع
        // معطيات
        // الدالة
        return $calculated_value;
    }
}
```

الخاتمة

في الختام احمد الله سبحانه وتعالي حمدا كثيراً طيباً مباركاً كما يليق بجلال وجهه وعظيم سلطانه ، أن وفقني لكتابة هذه الأسطر عن هذا الموضوع ، فان أصبت فمن الله وحده ، وان أخطأت فمن نفسي ومن الشيطان . وأتمنى أن أكون قدمت ما هو مفيد حول المعايير القياسية لكتابة الشيفرة المصدرية . وفي حال وجود ملاحظات أو أخطاء في هذا الإصدار يسعدني استقبالها على البريد الإلكتروني **majed@modernsys.net** .

كما أتقدم بجزيل الشكر والعرفان لمؤسسة دبليو ثري العربية الغير ربحية لما تقدمه من تعليم مجاني للغات برمجة وتصميم تطبيقات الويب للمستخدم العربي ، وما تنشره من مواضيع ودروس قيمة لإثراء المحتوى العربي.

وآخر دعوانا أن الحمد لله رب العالمين ، وصلى اللهم على نبينا محمد وعلى آله وصحبه أجمعين

المراجع

- Pear Coding Standards.
- zend framework wiki - PHP Coding Standard (draft).